Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема «Розробка адаптивної системи дистанційного навчання мові програмування C#»

Спеціальність - 121 “Інженерія програмного забезпечення”

Виконав студент

ІПЗ-41_____/ Дмитро КУПІН

(шифр групи)(підпис)(дата)(розшифровка підпису)

Науковий керівник

к.ф.-м.н._____/Сергій ПОЛЯКОВ

(посада) (підпис) (дата) (розшифровка підпису)

Консультант з питань нормоконтролю

фахівець._____/Тамара ЧАПОВСЬКА

Допускається до захисту

з питань нормоконтролю

Завідувач кафедри

д.т.н., проф._____/Олексій БИЧКОВ

(посада) (підпис) (дата) (розшифровка підпису)

Київ – 2021

Рішенням Екзаменаційної комісії

випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова екзаменаційної комісії

професор, д.т.н. Андрій БОНДАРЧУК

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ

СТУДЕНТУ

Купіну Дмитру Олександровичу

(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи

"Розробка адаптивної системи дистанційного навчання мові програмування C#"

керівник проекту (роботи) Поляков Сергій Анатолійович, к.ф.-м.н.

затверджена наказом вищого навчального закладу від „__"_____20__р. №_____

2. Строк здачі студентом закінченої роботи _____

3. Вихідні дані до роботи Теоретичні концепції та формальні моделі побудови та функціонування інформаційних та програмних технологій певного класу

4. Зміст пояснювальної записки (перелік питань, що їх належить розробити)

1. Аналіз предметної області

2. Аналіз призначення й мета створення мобільного додатку

2. Постановка задачі та методи дослідження

3. Моделювання мобільного додатку

4. Розробка мобільного додатку

5. Тестування мобільного додатку

5. Перелік графічного матеріалу (з точним забезпеченням обов'язкових креслень)

1. Статистика популярності мов програмування на березень 2021 (dou.ua) (рис. 1.1, ст. 13)
2. Світовий рейтинг мов програмування за березень 2021 (tiobe.ua) (рис. 1.2, ст. 13)
3. Сторінка вибору мови програмування в мобільному додатку "Sololearn" (рис. 1.3, ст. 15)
4. Основні сторінка мобільного додатку " Coding Quiz" (рис. 1.4, ст. 16)
5. Головна сторінка мобільного додатку " Programming Quiz" (рис. 1.5, ст. 17)
6. Статистика користувачів мобільних ОС (рис. 2.1, ст. 20)
7. Діаграма Ганта в візуальній формі (рис. 2.2, ст. 23)
8. Схематичне зображення екрану вибору складності тесту (рис. 3.1, ст. 28)
9. Проходження тестування (рис. 3.2, ст. 29)
10. Результат тестування (рис. 3.3, ст. 29)

Форма завдання на випускну кваліфікаційну бакалаврську роботу (на звороті першого аркуша)

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

Керівник _____ _____

(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ _____

(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Дослідження предметної області	04.03.21	виконано
2	Аналіз функціональних вимог	14.03.21	виконано
3	Аналіз нефункціональних вимог	24.03.21	виконано
4	Розробка алгоритмічної моделі	02.04.21	виконано
5	Розробка мобільного додатку	17.04.21	виконано
6	Тестування	07.05.21	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	.06.2021	виконано

Студент – бакалавр _____

(підпис) (розшифровка підпису)

Керівник роботи _____

(підпис) (розшифровка підпису)

АНОТАЦІЯ (на трьох мовах)

Випускна кваліфікаційна бакалаврська робота: 68 с., 27 рис., 11 табл., 2 додат., 14 джерела.

Тема: розробка адаптивної системи дистанційного навчання мові програмування C#

Об'єкт дослідження: дані найпоширених мов програмування, зацікавленність користувача та форма реалізації проекту.

Мета роботи: аналіз предметної області, цільової аудиторії проекту та проектування програмного забезпечення.

Предмет дослідження: технологія дистанційного навчання, мотивація в навчанні, форми та способи дистанційного навчання.

Результати дослідження:

Досліджено можливості застосування концепцій дистанційного навчання татеорії щодо реалізації проекту. Запропоновано формально-логічний підхід до мотивації в навчанні.

Висновок

В результаті досліджень було отримано дані, щодо цільової аудиторії проекту, теорії для впровадження та використання програмного забезпечення в реальних умовах.

ANNOTATION (in three languages)

Final qualifying bachelor's thesis: 68 p., 27 pics., 11 tables., 2 adds, 14 sources.

Topic: Development of an adaptive distance learning system in the C # programming language

Object of research: data of the most common programming languages, user interest and the form of project implementation.

Purpose: Analysis of the subject area, target audience of the project and software design.

Subject of research: distance learning technology, motivation in learning, forms and methods of distance learning.

Results of the research: Possibilities of application of concepts of distance learning and theory concerning project realization are investigated. A formal-logical approach to motivation in learning is proposed.

Conclusion

As a result of the research, data were obtained on the target audience of the project, theories for the implementation and use of software in real conditions.

АННОТАЦИЯ (на трех языках)

Выпускная квалификационная бакалаврская работа: 68 с., 27 рис., 11 табл., 2 доп., 14 источники.

Тема: Разработка адаптивной системы дистанционного обучения языку программирования C #

Объект исследования: данные наиболее распространенные языков программирования, зацікавленність пользователя и форма реализации проекта.

Цель работы: Анализ предметной области, целевой аудитории проекта и проектирования программного обеспечения.

Предмет исследования: технология дистанционного обучения, мотивация в обучении, формы и способы дистанционного обучения.

Результаты исследования: Исследованы возможности применения концепций дистанционного обучения и теории по реализации проекта. Предложено формально-логический подход к мотивации в обучении.

Заключение

В результате исследований было получено данные, по целевой аудитории проекта, теории для внедрения и использования программного обеспечения в реальных условиях.

Зміст

Оглавление

Завдання на дипломну роботу

В даному дипломному проекті була поставлена задача розробити адаптивної системи дистанційного навчання мові програмування C#.

Метою розробки даної системи є полегшити навчання студентам перших курсів. Умови роботи даного комплексу - наявність комп'ютера-сервера, на якому відбувається обробка всієї інформації, її наочне відображення і зберігання. Системні вимоги для роботи системи; для комп'ютера-сервера - наявність IBM PC сумісного комп'ютера, з операційною системою Windows 10.

Дипломна робота містить аналіз предметної області, аналіз функціональних вимог та не функціональних вимог. Було схематично розроблено прототип додатку та користувацький інтерфейс. З даних макету екранів додатку було розроблено візуальну частину додатку. Підключено систему аутентифікації користувача через Firebase Auth та встановлено функціональну частину додатку. Проведений етап тестування мобільного додатку для знаходження помилок згідно отриманих даних тестування додатку було виправлено всі знайдені помилки.

Пояснювальна записка складається зі вступу, 5 розділів, висновків, списку використаних джерел із 14 найменувань, додатків. Загальний обсяг 82 сторінок, у тому числі 52 сторінок основного тексту, 2 сторінки списку використаних джерел, 27 сторінок додатків.

Вступ

Представлена тема вважається актуальною і донині, адже вже мільйони людей вчаться і працюють щодня, не виходячи з дому.

У світі, і особливо в Україні, стрімко зростає кількість користувачів Інтернету і, відповідно, кількість користувачів дистанційного навчання.

При запуску проекту встановлюється, що програмний продукт повинен обслуговувати користувача під час тестування.

Метою проекту є створення мобільного додатку для проходження тестів декількома мовами програмування. Мотивуючим фактором для ідеї створення продукту стає його актуальність в самій тестовій системі її предмет тестів на мовах програмування.

В ході проходження робіт було встановлено і зафіксовано: мета проекту, завдання, які буде виконувати додаток, Вибір технологічних методів, за якими буде здійснюватися розробка і планування робіт – обрані методи планування графіка за методом Ганта. В рамках моделювання мобільного додатку були проаналізовані цільова аудиторія, сценарій взаємодії клієнт-додаток. Схематично побудований прототип і структура проходження основних екранів. Описує архітектуру мобільного додатку. Наступним етапом була фаза впровадження на рівні програмного забезпечення, де була розглянута основна функціональність програми. Після створення програми була проведена фаза тестування, яка перевірила всі можливі взаємодії користувача з програмним продуктом.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Розглядаючи питання системи або процесу навчання, ми не стикаємося з приємними проблемами, які не вирішують все правильно або взагалі не починають долати важкі моменти навчання. Основна проблема полягає не в ефективному використанні навчального часу і не в якісному засвоєнні теоретичного або практичного матеріалу. "Сам по собі досвід нічому не вчить... Без теорії досвід не має сенсу. Без теорії немає

питань, які можна було б задати. Тому без теорії немає навчання" – цитата американського вченого, професора Едварда Демінга, з яким згоден кожен досвідчений професіонал своєї справи, тому певний результат засвоєння теоретичного матеріалу є важливим і необхідним етапом навчання, і після проходження певного часу відповідного тесту знання почнуть залишатися усвідомленими, а для остаточного висновку по предмету потрібен метод проходження тесту, який є одним з основних допоміжних.

Було проведено аналіз популярності мов програмування на порталі dou.ua (рис 1.1) Графік складений з відгуків програмістів, які живуть в Україні та пишуть відповідними мовами у робочий час.

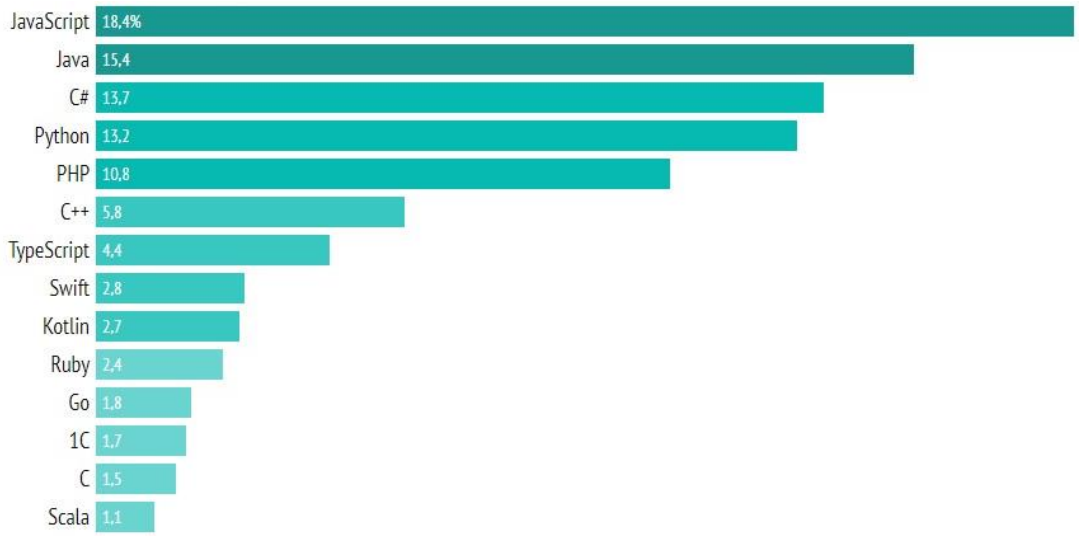


Рисунок 1.1 – Статистика популярності мов програмування на березень 2021 (dou.ua)

Рейтинг від tiobe.com - світовий рейтинг мов програмування за березень 2021 (рис. 1.2) Сервіс збирає інформацію про популярність мов програмування в різних країнах і формує щомісячні рейтинги. Дані засновані на кількості курсів, думках досвідчених програмістів і запитах в Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube і Baidu.

Apr 2021	Apr 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	14.32%	-2.40%
2	1	↓	Java	11.23%	-5.49%
3	3		Python	11.03%	+1.72%
4	4		C++	7.14%	+0.36%
5	5		C#	4.91%	+0.16%
6	6		Visual Basic	4.55%	-0.18%
7	7		JavaScript	2.44%	+0.06%
8	14	⇈	Assembly language	2.32%	+1.16%
9	8	↓	PHP	1.84%	-0.54%
10	9	↓	SQL	1.83%	-0.34%

Рисунок 1.2 – Світовий рейтинг мов програмування за березень 2021 (tiobe.ua)

Зібравши відповідну статистику бачимо стабільність в популярності мови програмування C# - на українському ринку. C# є лідером для професійних розробників та студентів. Тому були обрано дану мову програмування, яка сьогодні займа є провідне місце серед програмістів і початківців. Для успішного вивчення теоретичного або практичного матеріалу ви завжди повинні закріплювати свої знання спочатку у формі тесту, а потім самостійно в середовищі розробки програмного забезпечення. Всім, хто знає всі мови програмування на базовому рівні або має певний досвід в розробці ПП, буде цікаво перевірити свої знання та отримати результат свого тесту. У даній роботі представлений проєкт мобільного додатку - тестування мови програмування, де користувач перевіряє свої навички та отримує результат проходження.

1.2 Аналіз мобільних додатків для проходження тестування

Під час планування проекту було розглянуто декілька популярних та подібних додатків з метою аналізу функціоналу продукту, користувацького інтерфейсу, виокремлення сильних факторів та знаходження недоліків.

Схожі мобільного додатку бувають з вмістом теоретичним матеріалом та проходження тестування, турнірні дуелі між користувачами додатка та тільки проходження тесту. Перший приклад один з популярних додатків Sololearn, зображений на рисунку 1.3, на тему тестування та освіти загалом, яка має велику базу даних теоретичного матеріалу та тестових завдань. Користувачі також можуть змагатися між собою хто більше відповість правильних питань той і переміг таким чином присутній дух змагання, який подобається користувачам. Недоліки додатку полягають у відсутності, частково, актуального контенту, що не відповідає сучасному стану розвитку мов програмування, що може заплутати початківців, присутність реклами, яку можна відключити тільки купляючи підписку.

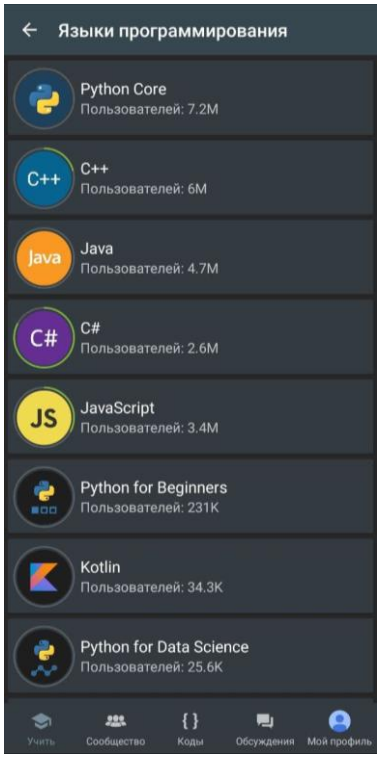


Рисунок 1.3 – Сторінка вибору мови програмування в мобільному додатку “Sololearn”

Наступний додаток Coding Quiz зображений на рисунку 1.4, влаштований тільки для тестування, представляє велику кількість питань з рівнем складності в тестовому форматі. Сильні сторони приємний та інтуїтивно зрозумілий інтерфейс та великий вибір різних мов програмування. Також додаток має присутність реклами, яка відключається за гроші. Присутня тільки англомовна версія додатку, тому хто не володіє буде не комфортно користуватися.

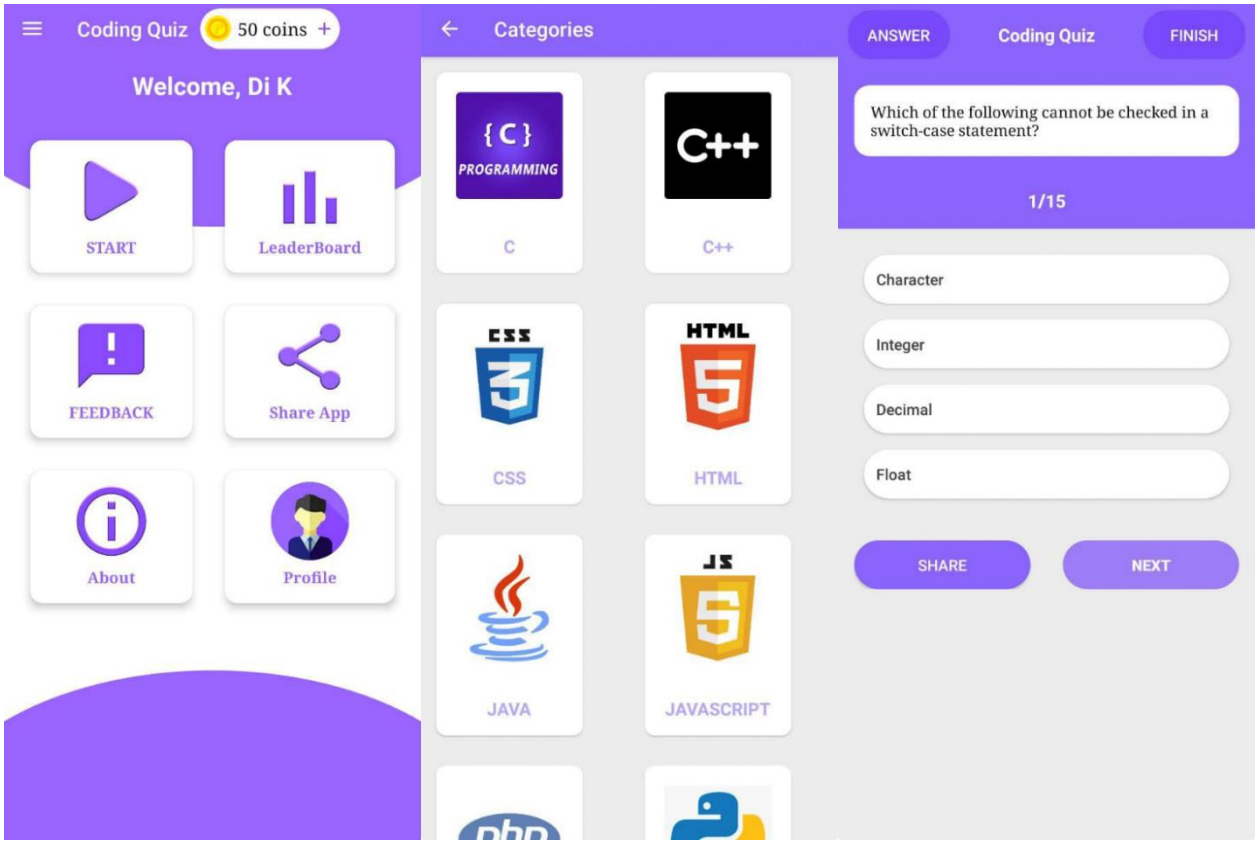


Рисунок 1.4 – Основні сторінка мобільного додатку “ Coding Quiz”

Додаток Programming Quiz зображений на рисунку 1.5 має широкий функціонал для вибору проходження тесту на більшість тем з комп'ютерних наук. Також продукт не має реклами, що є позитивним чинником для користувачів. Після проходження тесту користувач отримує результат проходження та відповідний коментар який залежить

від оцінки складання тесту. Згідно отриманих даних було складено порівняльну таблицю 1.1

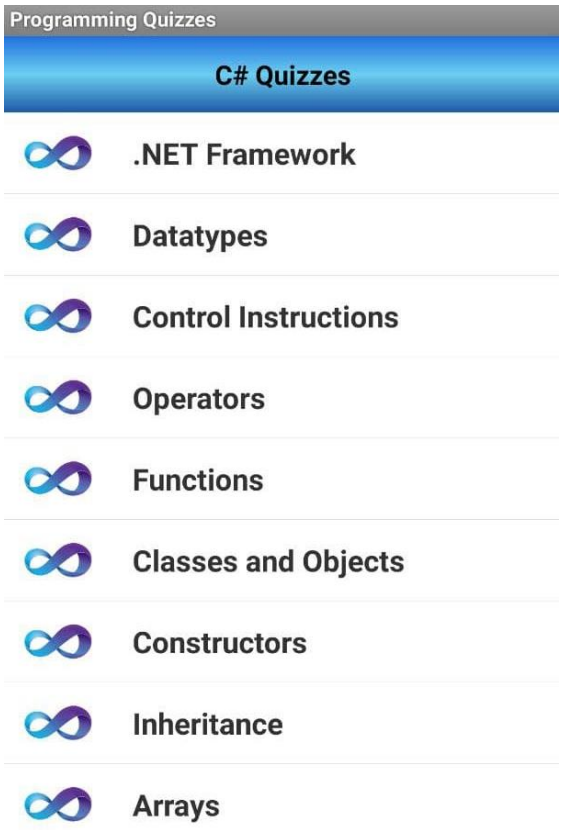


Рисунок 1.5 – Головна сторінка мобільного додатку “ Programming Quiz”

Зробивши відповідні підсумки на таблиці 1.1 продемонстровано основні переваги мобільного додатка. Схожі додатки мають свою мету, задачі та цільову аудиторію тому вони можуть відрізнятися, але мета порівняння знаходження позитивних властивостей та характеристик, які потрібно уникнути при створенні ПП, що зробить продукт ціннішим серед потенційної аудиторії.

Характеристика	Sololearn	Coding Quiz	Programming Quiz
Зручний інтерфейс	+	+	-
Мови користування українська	-	-	-
Сучасний дизайн	+	+	-

Відсутня нав’язлива реклама	-	-	+
Можливість обрання рівня складності	-	+	-
Кількість завантажень в Google Play	10 000 000+	100+	1000+

Таблиця 1.1 – Порівняння мобільних додатків зі схожим функціоналом

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1 Задачі програмного продукту

Сьогодні велика кількість людей вважають за необхідне знайти час для вдосконалення своїх освітніх навичок і придбання нових знань для власних потреб з метою самоосвіти, яке є результатом досягнення успіху в професійній діяльності. Кожен вибирає свій спосіб використання інформації, отриманої з підручника, мережі або від викладача, але для закріплення і вдосконалення нових знань необхідно пройти тест з відповідної, спеціалізованої теми. Тому створення мобільного додатку зі зручним інтерфейсом для проходження тестів на мовах програмування є вирішальним фактором для самоосвіти користувача.

Повний перелік функціональних вимог наведено в таблиці 2.2

- Виберіть кілька варіантів складності з мовами програмування (Легкий, Середній, Важкий).
- Наявність "життів", після втрати яких тест закінчується відповідним коментарем і оцінкою.
- Після проходження тесту можливість перегляду правильних і неправильних відповідей.

Корисний додаток для проходження тесту, воно буде містити профільні питання і можливі відповіді, з яких тільки один правильний. Основний функціонал - це вибір вибір складності тесту та звичайно ж сам процес проходження тесту і вже в результаті отримання прохідного бала. Під час тесту у користувача є можливість вибрати правильний варіант, який він вважає за необхідне, і вийти з процесу тестування до вибору складності.

Програма допоможе вам перевірити знання користувачів на різних рівнях складності з мови програмування C#, тим самим мотивуючи вас підвищити свої знання або перевірити правильність відповіді, що в будь-якому випадку зробить корисний вплив на користувача.

2.2 Вибір методів

Після проведення аналізу актуальності роботи можна зазначити про популярність мови програмування Kotlin, яка займає першу сходинку за рейтингом tiobe.com та є одним з кращим інструментом для написання Android додатків. Більшість смартфонів працює на базі операційної системи Android це підтверджує статистика від statcounter.com (рис. 2.1), тому було вирішено писати додаток для Android користувачів. Досвідчені розробники від Google пропонують користуватися середовищем Android Studio, яка бездоганно підтримує мову Kotlin, має емулятор для тестування додатка та систему контроля версій Git.

Рисунок 2.1 Статистика користувачів мобільних ОС

2.3 Планування робіт

Виокремлення мети проекту методом SMART

SMART-цілі - метод постановки конкретних цілей, реалістичних завдань, які чітко обмежені часі та мають свій кінцевий термін та аналіз роботи виконавця якийзнає скільки роботи вже виконано.

Методологія SMART являє собою концепцію правил:

- Конкретність (Specific) - прозора постановка мети, де ціль повинна бути зрозумілою.
- Вимірюваність (Measurable) – виконавець має уявлення на якому етапі йде розробкапроекту
- Досяжність (Achievable) – мета конкретної задачі закріплена за одним виконавцем якийповинен розуміти всі критерії цілі та нести відповідальність.
- Реалістичність (Relevant) – мета повинна мати свої певні реальні потреби, які залежатьвід виконавця.
- Обмеженість в часі (Time-bound) – проект має бути виконаний в певний, заданий час.
- Кожен етап проекту потрібен мати свій ліміт часу.

Результати розміщені у таблиці 2.1

Specific (Конкретність)	Мобільний додаток для проходження тестування з мов програмування
Measurable (Вимирюваність)	Результатом роботи є відгуки користувачів, які надали свою оцінку
Achievable (Досяжність)	Реалізація виконана на мові програмування Java, для користувачів ОС Android
Realistic	В наявності є всі необхідні технічні та програмні засоби.

(Актуальність)	
Timed (Обмеженість в часі)	Ціль має свій кінцевий термін. Кожен етап повинен бути виконаний вчасно. Терміни повинні бути обговорені виконавцем та керівним проекту або замовником.

Таблиця 2.1 – Висновок мети методом SMART

Планування змісту структури робіт проекту (WBS)

WBS — ієрархічна структура, необхідна для логічного детального розподілу усіх робіт на окремі операції, які плануються відповідно до обраної моделі життєвого циклу. Робота дерева проекту WBS мусить бути поділена на головні задачі та під задачі.

Переваги використання WBS:

- Чітка структуризація проекту;
- Надає допомогу в описі змісту проекту для зацікавлених сторін;
- Розподіл обов'язків серед команди проекту;
- Наглядно описує конкретні етапи проекту

Створена WBS-діаграма представлена на схемі Б.1 у Додатку Б

Побудова матриці відповідальності

Після побудови структур WBS формується матриця відповідальності за проект, і на кожному етапі проекту повинен бути відповідальний член проектної команди. Етап роботи не обов'язково повинен виконуватися одним членом команди, їх може бути кілька, головне - бути відповідальним, хто контролює і відповідає за процес роботи та результат. Якщо на певному етапі є кілька виконавців, відповідальна особа призначає їх Виконавцю і помічникам або консультантам.

Матриця відповідальності представлена у таблиці Б2 у Додатку Б.

Побудова календарного графіку виконання

Побудова календарного графіка реалізації проекту, безумовно, є важливою частиною, тому у розробника є візуальний план дій про тривалість робіт. Використовуючи діаграму Ганта, Розробник бачить терміни виконання кожного завдання через те, що багато завдань проекту часто залежать один від одного, може бути важко розрахувати, скільки часу має бути виділено на завдання, коли саме вона повинна бути запущена і до якого часу повинна бути завершена. Стовпці на діаграмі Ганта використовуються в якості індикатора часу, який необхідно виділити для виконання завдань, і таким чином Ви отримуєте більш широку і детальну картину всього проекту і термінів виконання його завдань.

Діаграма Ганта була складена з використанням excel в некомерційних цілях. Проект розробки додатків був розділений на шість частин, кожна частина містити свої власні завдання, які мають свою тривалість в днях. Загалом, кожна задача досить унікальна, тому деякі завдання виконувалися по-різному за тривалістю. Кожне завдання мати (в собі) стан виконання і завершення, тому між ідентичними завданнями існують логічні зв'язки. Всі шість частин проекту повинні бути узгоджені та завершені в строк.

На рисунку 2.2 представлена діаграма Ганта

Дата	**Етапи**	**Зміст**
04.03.21 – 14.03.21	Дослідження предметної області	Визначення мети роботи і задач, попередній аналіз витрат, необхідних навичок та програмного забезпечення.
14.03.21 – 23.03.21	Аналіз функціональних вимог	Ідентифікація дій, які система повинна бути здатною виконати, зв'язок входу / виходу в поведінки системи.
24.03.21 – 01.04.21	Аналіз нефункціональних вимог	Визначення критеріїв роботи системи в цілому.

02.04.21 – 16.04.21	Розробка концепції дизайну	Уточнення структури вхідних і вихідних даних, визначення форми їх уявлення та створення прототипу
17.04.21 - 06.05.21	Розробка мобільного додатку	Процес реалізації мобільного додатку.
07.05.21 – 17.05.21	Тестування	Початок тестування написаного коду, перевірка коду.

Таблиця 2.2 Календарний графік роботи проекту

Рисунок 2.2 Діаграма Ганта в візуальній формі

Управління вимогами

Функціональні вимоги - це опис дій програмного продукту. Він описує програмну систему або її компонент. Наприклад, ви можете взаємодіяти з користувачем або запитувати дані Користувача. Функціональні вимоги також називаються функціональними специфікаціями.

Таблиця 2.3 – Список функціональних вимог

№	Вимога
1	ПП повинен відображати головний екран
2	ПП повинен відображати список всіх рівнів складностей
3	ПП повинен відображати панель навігації
4	У ПП повинен бути передбачена вихід з тестування

5	Користувацький інтерфейс ПП повинен бути інтуїтивно зрозумілим для користувача
6	У ПП повинно знімати “ життя ” якщо користувач не правильно відповідає на питання тесту:
7	ПП повинно припиняти тестування якщо користувач відповів не правильно 3 рази
8	ПП повинний показувати результат проходження тесту(кількість правильних балів, не правильних та пропущених)

Нефункціональні вимоги (NFR) визначають атрибут якості програмної системи. Вони оцінюють програмну систему на основі чутливості, зручності використання, безпеки, переносність та інших нефункціональних стандартів, які мають вирішальне значення для успіху програмної системи. Недотримання нефункціональних вимог може призвести до того, що системи не відповідатимуть потребам Користувача. (таблиця. 2.4)

№	Вимога
1	ПП має підтримувати ОС Android 8.1
2	ПП повинен бути виконаний англійською мовою.
3	Реалізація ПП відбувається мовою Kotlin у середовищи розробки Android Studio

Таблиця 2.4 – Список нефункціональних вимог

Управління ризиками

Ідентифікація ризиків - заздалегідь оцінити та проаналізувати всі можливі ризики проекту і структуровано описати їх і їх наслідки. У складанні такого списку

ризиків беруть участь керівник проекту, старший Розробник, архітектор проекту, замовник і команда аналітиків. Якісна оцінка ризиків - це процес представлення якісного аналізу ідентифікації ризиків та виявлення ризиків, що потребують швидкого реагування. Ця оцінка ризику визначає важливість ризику та вибирає метод реагування.

Якісна оцінка ризиків може бути проведена за допомогою загального методу аналогій, суть якого полягає в аналізі набору даних за аналогічними проектами. Тут вони проводять дослідження впливу на них несприятливих факторів, щоб точно визначити потенційний ризик реалізації нових проектів.
"Основна складність використання методу аналогій полягає в точному і правильному виборі аналогів, бо немає формальних критеріїв для визначення ступеня подібних ситуацій".

Таблиця 2.5 – Матриця ризиків

Оцінка	**Ймовірність виникнення:**	**Величина втрат:**
1	Вірогідність мінімальна	Мінімальна
2	Практично малоймовірно	Низька
3	Можливе виникнення на достатньому рівні	Середня
4	Виникнення загрози скоріше ймовірне	Висока
5	Вірогідність досить близька до катастрофи	Максимальна

План по усуненню ризиків:

- Відкритість та адекватна робота з замовником проекту
- Взаємодія з замовником заздалегідь вказаними датами
- Прототипування і узгодження інтерфейсу
- Поставки замовникам результати виконаних задач

- Надійний вибір інструментів виконання проекту

3. МОДЕЛЮВАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1. Збір вимог

Цільова аудиторія

Результатом цього проекту є мобільний додаток, який служить помічником для користувачів, які хочуть перевірити свої навички в проходженні тестів на мові програмування C#. Користувач може вибрати складність розділу та приступити до проходження тесту в кінці проходження, користувач отримує результат з відповідною оцінкою. Цільова аудиторія – це студенти перших курсів, зацікавлені люди, які люблять програмувати, тому проходження тестів по відповідній темі буде цікавим і корисним.

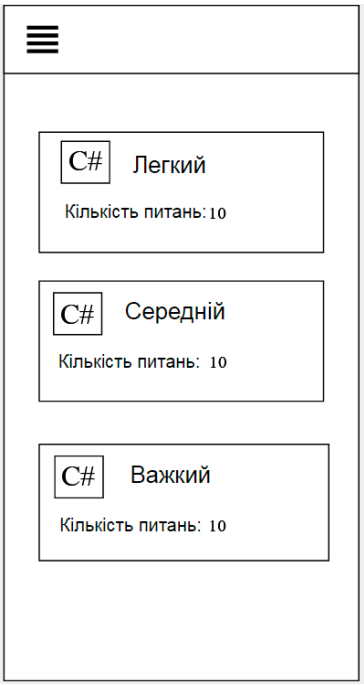
Сценарії взаємодії

Після відкриття Програми Користувач бачить стартову сторінку з головним екраном, де він може вибрати відповідний рівень складності, а потім почати тестування. Під час проходження користувач відразу бачить кількість "життів", після втрати їх усіх тестування завершено. Коли ви натискаєте на відповідний варіант відповіді, користувач відразу ж бачить, чи правильно він відповів чи ні. Правильна відповідь не буде знімати життя, а неправильна зніме одне життя. Таким чином, після завершення проходження користувач потрапляє на екран результатів, відразу ж показує кількість правильних і неправильних відповідей, коментар до результату тесту. Після перегляду результату Користувач також може повторно пройти тест, натиснувши на відповідну кнопку.

3.2 Проектування прототипу та структура проходження додатка

На цьому етапі моделювання програмного продукту вказується прототип екранів взаємодії з користувачем - це необхідно для установки основних елементів інтерфейсу для вже більш простого проектування дизайну користувальницького інтерфейсу. Ця модель використання

прототипу додатка використовується при отриманні інформації від замовника, де він вказує основні потреби інтерфейсу і всі варіанти взаємодії між додатком і користувачем, які йому потрібні, і після узгодження прототипу проект переходить до етапу створення користувальницького інтерфейсу. Структура основної частини головного екрану являє собою вибір



рівня складності тесту для мови програмування C#.

Рисунок 3.1– Схематичне зображення екрану вибору складності тесту

Потім був розроблений екран для проходження тесту. (рис. 3.2). Вам потрібно звернути увагу на можливість виходу з тесту, кількість "життів" користувача, тестові питання, варіанти відповідей. Та кінцевий етап тестування - це результат.(рис. 3.3)

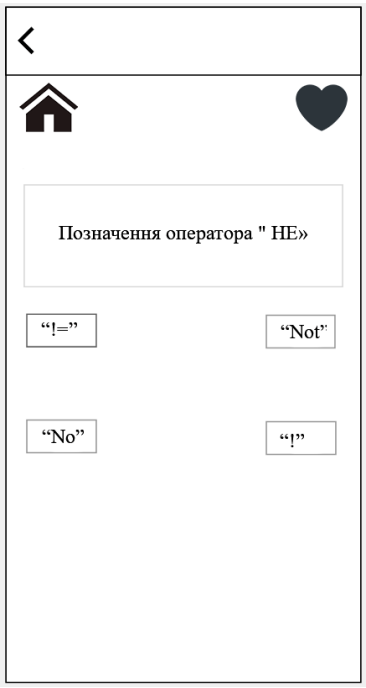


Рисунок 3.2 – Проходження тестування

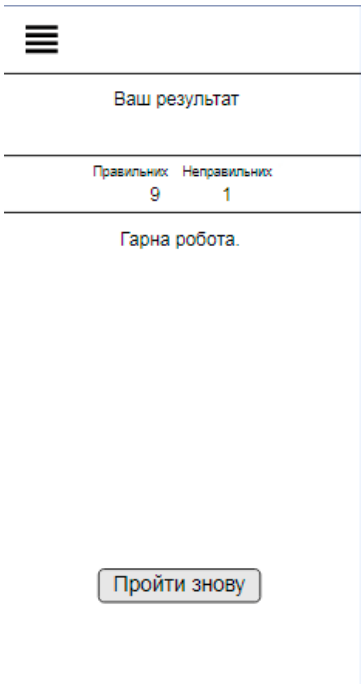


Рисунок 3.3 – Результат тестування

Діаграма взаємодії

Діаграма варіантів використання - це список дій, сценарій, в якому користувач взаємодіє з додатком або програмою для виконання дії для досягнення певної мети.

Варіанти використання були вказані в тексті та представлені діаграмою варіантів використання. Компонент сутності моделювання діаграми взаємодії допомагає розробити систему з точки зору майбутнього користувача. Використовуючи прецедент, ви можете описати вимоги користувача, вимоги до взаємодії з системою та опис взаємодії людей і компаній в реальному житті.

Була створена діаграма варіантів використання, яка показувати відносини між суб'єктами та варіантами використання і є невід'ємною частиною моделі варіантів використання. Діаграма була побудована з використанням UML і сайту draw.io (рис. 3.6).

Параметри використання Програми для авторизованих користувачів:

- Виберіть розділ для проходження тесту.
- Виберіть рівень складності.
- Отримати результат і коментар.
- Редагування змісту самих тестових питань і можливих відповідей.
- Перегляд і редагування призначених для користувача даних програми

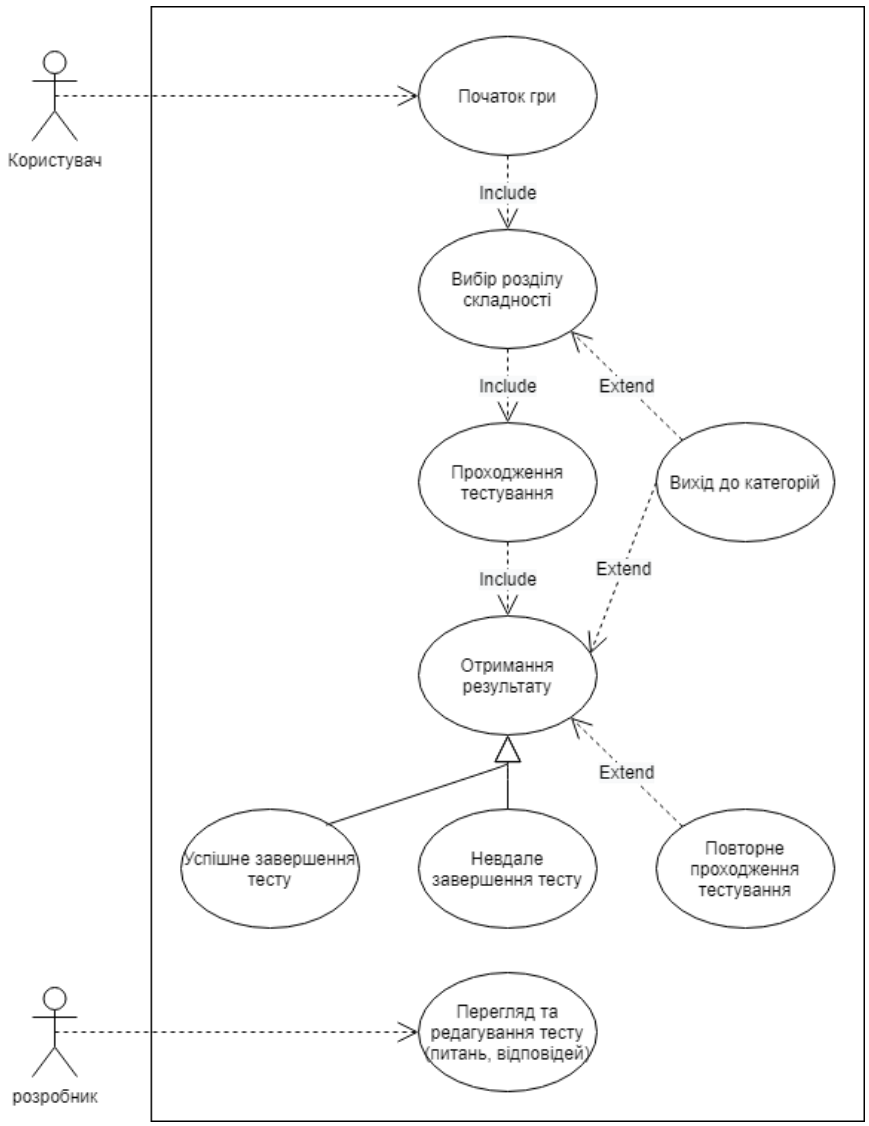


Рисунок 3.6 – Діарама прецедентів

Діаграма послідовності — різновид діаграми в UML. Діаграма послідовності відображає взаємодії об'єктів впорядкованих за часом. Зокрема, такі діаграми відображають задіяні об'єкти та послідовність відправлених повідомлень. Діаграма була побудована з використанням UML і сайту draw.io (рис. 3.7).

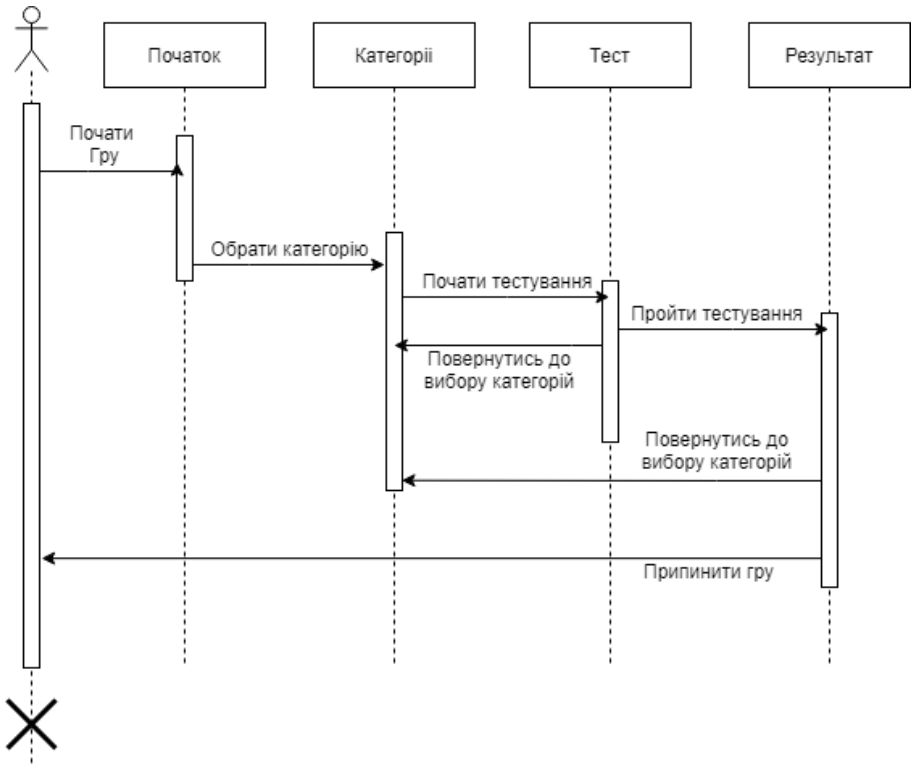


Рисунок 3.7 – Діарама послідовності

Діаграма компонентів (Component diagram) описує фізичне представлення системи та забезпечує перехід від логічного представлення до реалізації проекту в формі програмного коду. Компонент є частиною фізичної реалізації системи (наприклад програмний модуль або інтерфейс користувача), який інкапсулює певний набір функціональних можливостей. Діаграма була побудована з використанням UML і сайту draw.io (рис. 3.8).

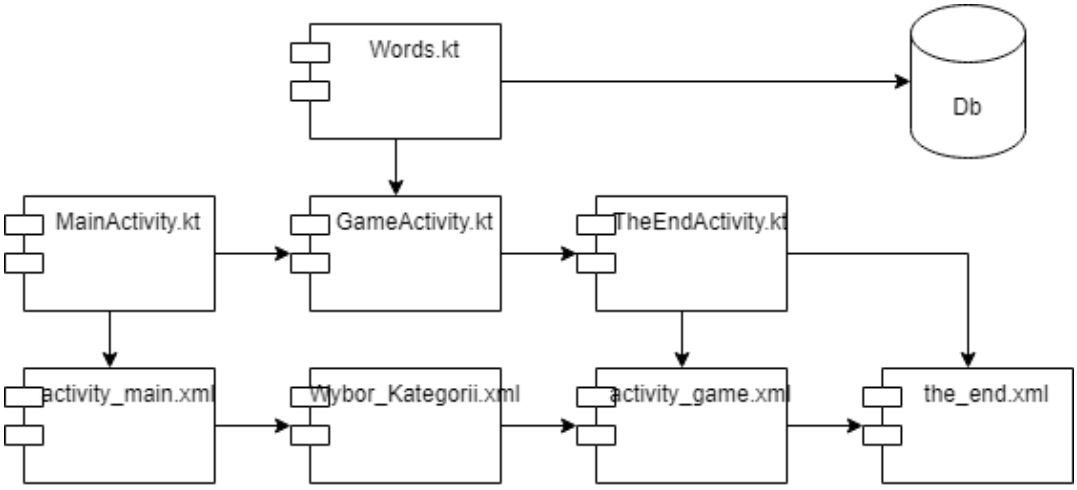


Рисунок 3.8 – Діарама компонентів

4. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1 Огляд основних можливостей програмного продукту

Розроблюваний мобільний додаток повинен забезпечувати наступний функціонал:

1. **Інтуїтивно зрозумілий інтерфейс**. Було створено заздалегідь макети екранів взаємодії з користувачем та підібрані відповідні стилі для приємного перегляду інформації додатку.
2. **Своєчасна швидкість продукту та відображення актуальної інформації**. Було застосовано сучасні засоби для обслуговування користувача.
3. **Вибір користувачем рівня складності.** Всі режими переходів між екранами додатка повинні працювати доцільно.
4. **Проходження тестування.** Користувачеві надається можливість обрати одну правильну відповідь та перейти на інше питання.
5. **Отримання результату проходження тесту.** Фінальний результат виглядає –кількість набраних балів та неправильних балів.

Програмний продукт створений у вигляді мобільного додатку. Основна перевага мобільних додатків що вони гнучкі до використання.

4.2 Розробка дизайну інтерфейсу

В даному розділі було спроектовано UI користувача де було підібрано основні кольори додатка проходження тесту на (рисунок 4.1) всі інші макети розробки булопредставлено в Додатку А

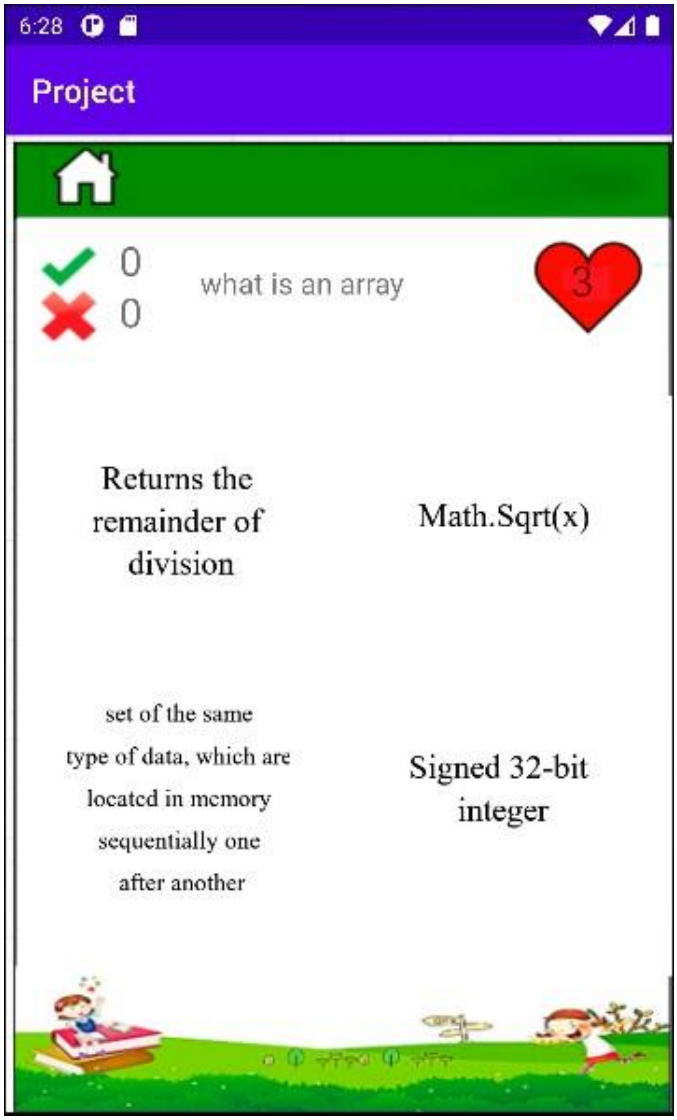


Рисунок 4.1 – Макет екрану проходження тестування

Користувач потрапляє на головну сторінку вибору рівня складності (рис.4.2)

Рисунок 4.2 Вибір рівня складності

Після обрання рівнів складності користувач потрапляє до етапу проходження тестування рис 4.3. На рис 4.4 бачимо, що вже друге питання та залишилосьдвоє “життів”.

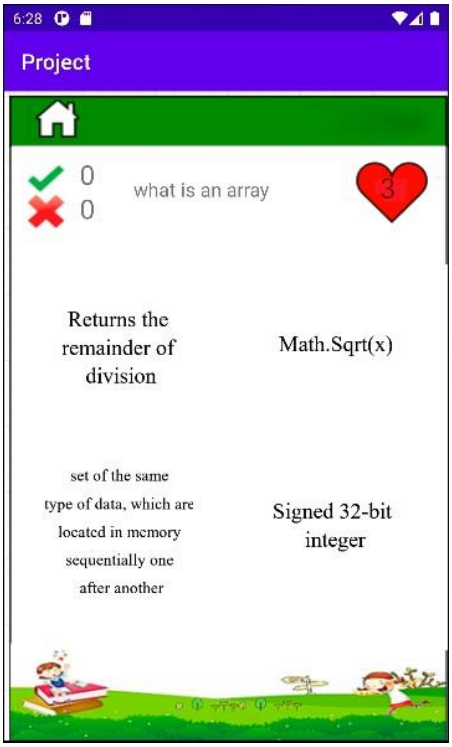


Рисунок 4.3 Вигляд екрану тесту

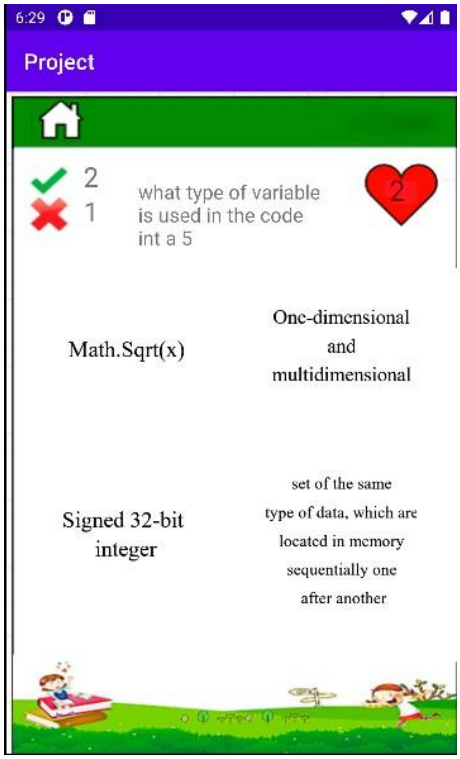


Рисунок 4.4 Обрання не вірного варіанту відповіді

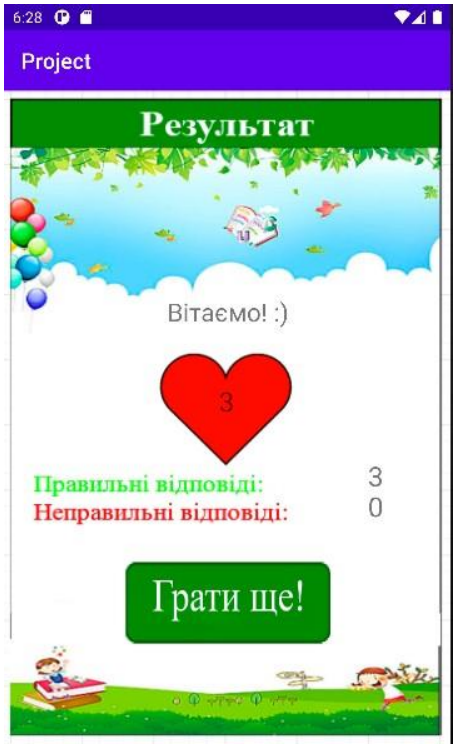


Рисунок 4.5 Отримання результату проходження тестування

На даному вікні спостерігаємо кількість правильних та неправильних відповідей та коментар додатку. Також присутня можливість почати тест заново та на (рис 4.6)

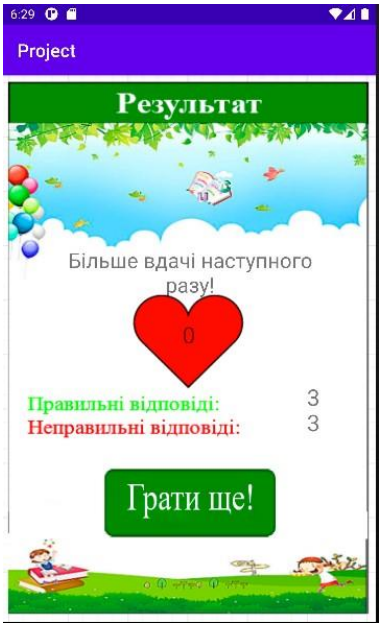


Рисунок 4.6 Отримання результату проходження тестування

5. ТЕСТУВАННЯ

Після закінчення етапу розроблення програмного продукту потрібно протестувати всі можливі варіанти взаємодії користувача з додатком. В таблиці 5.1 продемонстровано детальний опис всіх тестів.

Таблиця 5.1 – Тестування мобільного додатку

TC_ID	Test Scenario/Test Step Name	Actions	ER	Статус
Маємо доступ до інтернету та додаток відкритий на эмуляторі смартфона Nexus 6				
TC1.1	Перевірити всі екрани складностей	На екрані з C# 1. Перевірити всі екрани складностей 2. Перевірити коректність данних	Унікальні питання для кожного тесту	+
TC1.2	Перевірка Правильності проходження тесту	1. Під час проходження потрібно щоб було встановлено кнопка виходу з тесту 2. Кількість життів повинна бути 3 та при не правильності вибору в тесті зменшуватись на одиницю 3. Питання повинно бути добре видно 4. Варіантів відповідей повинно бути 4 5. Правильна відповідь повинна бути одна 6. Перевірити щоб працювати кнопка пропустити питання	Після натискання кнопки виходу показує вікно для уточнення виходу зтесту. Кількість життів працює коректно. Питання видно відмінно. Після вибору варіанта відповідіпоказує відповідних колір зелений – правильна відповідь червонане правильна. Після натискнення кнопки пропуску питання з’являється попереджувальне вікно про пропуск тесту	+

Продовження таблиці 5.1 – Тестування мобільного додатку

ТС1.3	Перевірка результату тесту	1. Після проходження тесту з'являється вікно результату 2. Натиснути почати тест знову "+" 3. Перевірити правильну кількість балів(набраних очокта не правильних балів, та пропущених)	Після натискання на пропуск тесту з'являється вікно з привітанням та в якому вказано минулу кількість балів у проходженні тестування Всі дані відображені правильно	+

ВИСНОВОК

Результатом кваліфікаційної роботи є дані, щодо цільової аудиторії проекту, теорії для впровадження та використання програмного забезпечення в реальних умовах, робочий мобільний додаток для перевірки знань мов програмування. Результати аналізу предметної області дозволили підтвердити актуальність роботи. На етапі аналізу порівняння нинішніх додатків з аналогічним функціональним призначенням були виявлені позитивні особливості відомих додатків і особливості, яких слід уникати при проектуванні. Дотримання визначених вимог до програмного продукту дозволило розробити конкурентно спроможний мобільний додаток, готовий до впровадження. Вибір методів реалізації був обраний за рекомендаціями від фахівців з провідної компанії Google.

Планування ІТ – проекту виконано за класичною методикою. Результати планування представлені у формі діаграми Ганта, діаграми WBS, таблицями результатів оцінювання ризиків тощо.

Під час моделювання проекту визначені кінцеві споживачі продукту та сценарій взаємодії користувача з додатком, розглянуті всі можливі сценарії взаємодії користувача з додатком в корисній та розважальній формі.

Макет додатку, розроблений засобами сайту moqus.com. Реалізація внутрішнього функціоналу проходження тестування реалізовано методом завантаженням відповідних даних з json далі відповідних класах activity триває обробку отриманих даних правильно опрацьовані було створено відповідний адаптер даних який зв'язує інтерфейс View частину та модель, яка в свою чергу за допомогою інтерфейсу

СПИСОК ЛІТЕРАТУРИ

1. ЦИТАТИ: Вільям Едвардс Демінг Веб-сайт URL http://www.management.com.ua/cit.php?author=%C2%B3%EB%FC%FF%EC%20%C5%E4%E2%E0%F0%E4%F1%20%C4%E5%EC%B3%ED%E3 (дата звернення 04.05.2021)
2. TIOBE Index for March 2021 Веб-сайт URL https://www.tiobe.com/tiobe (дата звернення 04.05.2021)
3. What is Work Breakdown Structure? Веб-сайт URL https://www.visual-paradigm.com/guide/project-management/what-is-work-breakdown-structure/ (дата звернення 06.05.2021)
4. Діаграма Ганта для Управління Проектами: Плюси і Мінуси. Веб-сайт URL https://blog.ganttpro.com/ru/gantt-chart-online-project-planning-pros-cons-ru/ (дата звернення 12.05.2021)
5. What is a Functional Requirement? Specification, Types, EXAMPLES Веб-сайтURLhttps://www.guru99.com/functional-requirement-specification-example.html (дата звернення 15.05.2021)
6. What is Non-Functional Requirement? Веб-сайт URLhttps://www.guru99.com/non-functional-requirement-type-example.html (дата звернення 16.05.2021)
7. Якісна оцінка ризиків. Веб-сайт URLhttp://ru.solverbook.com/spravochnik/menedzhment/kachestvennaya-ocenka-riskov/ (дата звернення 18.05.2021)

8. Архітектура мобільного клієнт-серверного додатка Веб-сайт URL https://habr.com/ru/post/246877/.(дата звернення 20.05.2021)
9. Що таке Юзкейс (Use Case) або "Сценарій Використання" в тестуванні ПО? Веб-сайт URL https://software-testing.org/testing/chto-takoe-yuzkeys-use-case-ili-scenariy-ispolzovaniya-v-testirovanii-po.html /.(дата звернення 23.05.2021)
10. USE CASES Що це таке і навіщо вони потрібні? Веб-сайт URL https://systems.education/use-case/ дата звернення (23.05.2021)
11. Методологія IDF0 Веб-сайт URL https://uk.wikipedia.org/wiki/IDEF0 (дата звернення 27.05.2021)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на Розробка адаптивної системи дистанційного навчання мові програмування C#

Продовження додатку А

1. Призначення й мета створення мобільного додатку.

1.1. Призначення мобільного додатку

Мобільний додаток призначений для перевірки знань користувачів з мови програмування C# з різним рівнем складності.

1.2. Мета створення додатку

Допомогти користувачам перевірити свої знання та подивитися свою кількість балів.

1.3. Цільова аудиторія

Аудиторія, яка знайома з програмуванням на мові C#

2. Вимоги програмного продукту

2.1. Вимоги до програмного продукту загалом

2.1.1. Вимоги до структури й функціонування

Мобільний додаток повинен бути в мінімалістичному стилі та інтуїтивно зрозумілим для користувачів, продукт здатен перевіряти користувачів у вигляді тестування та показувати набранні бали. Кожен тест має 3 “життів” втративши їх користувач закінчується тест з негативною оцінкою.

2.1.2. Вимоги до користувачів та персоналу

Для користуванням додатком потрібен відповідний apk файл, який можна встановити на смартфон на базі Android версією не менше 8.1

2.1.3. Вимоги до подання інформації

Вся основна інформація тестування є офлайн та користувач має змогу проходити тестування офлайн.

2.1.4. Основні вимоги

2.1.4.1. Структура додатку

Додаток повинен складатися з наступних екранів користувача:

- Головний екран – користувач обирає складність тесту.
- Екран проходження тестування – користувач читає питання та обирає правильний варіант відповіді після чого переходить на інше питання.
- Екран результату тестування – користувач після закінчення тесту дивиться на кількість правильних, не правильних та пропущених питань. Також має змогу побачити на правильні відповіді на питання, які він відповів не правильно. Та можливість повторно пройтитестування.

2.1.4.2. Навігація

Інтерфейс додатку повинен бути інтуїтивно зрозумілий і мати досить простийфункціонал, розділи додатку повинні бути доступні для користувача. Система повинна забезпечувати навігацію по всіх доступних користувачеві розділам і відображати відповідну інформацію.

2.1.4.3. Система навігації (дизайн головного екрану додатку)

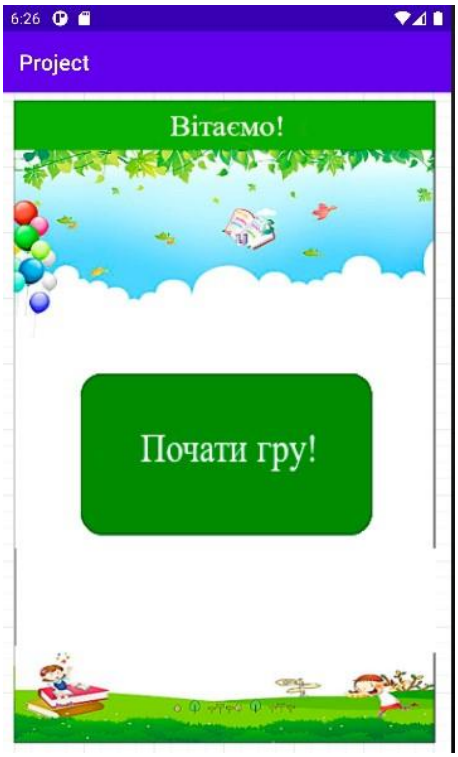


Рисунок А.1 – Дизайн головного екрану додатку

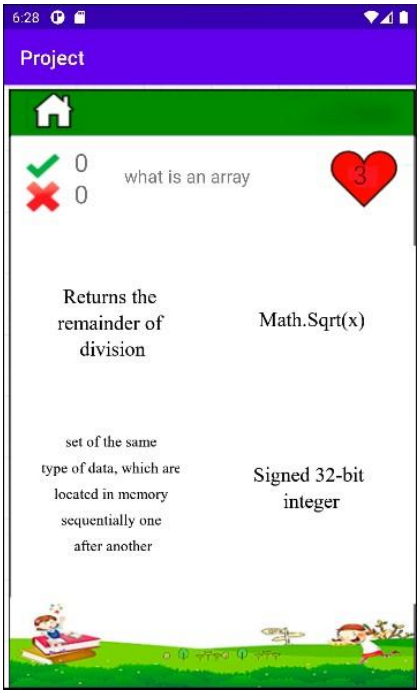


Рисунок А.2 – Дизайн проходження тестування екрану додатку

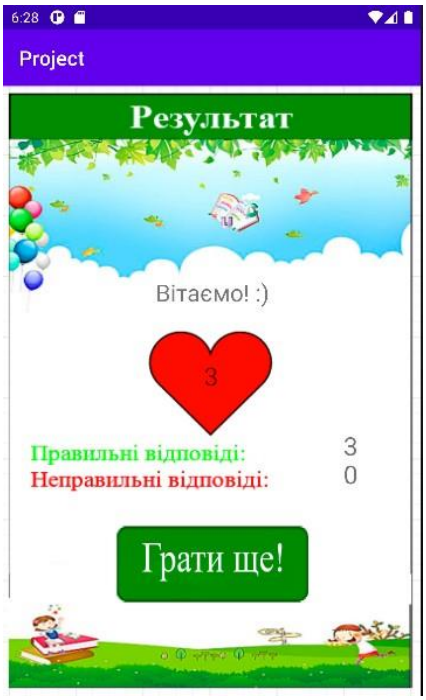


Рисунок А.3 – Дизайн результату проходження

2.1.4.4. Типові навігаційні й інформаційні елементи

- Верхня частина додатку(тулбар)
- Основне поле для вибору розділа тестування

2.2. Вимоги до видів забезпечення

2.2.1. Вимоги до інформаційного забезпечення

Реалізація програмного продукту відбувається за допомогою мови програмування Kotlin. В середовищі Android Studio

2.2.2. Вимоги до лінгвістичного забезпечення

Додаток повинен бути виконаний англійською мовою.

2.2.3. Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Наявність смартфону на базі Android версія не менше 8.1

3. Склад і зміст робіт

Докладний опис етапів роботи зі створення додатку наведено в табл. А1.

Таблиця А1 – Етапи створення функціонального додатку

№	Склад і зміст робіт	Строк розробки (у робочих днях)
1	Дослідження предметної області	10 днів
2	Створення дизайн макетів екранів додатка	10 днів
3	Верстка екранів	5 днів
4	Створення функціоналу додатку	20 день
5	Тестування та знаходження несправностей. Виправлення недоліків	10 днів

Для створення умов функціонування, при яких гарантується відповідність створюваного додатку вимогам сьогодення ТЗ і можливість його ефективної роботи,повинен бути проведений певний комплекс заходів.

ДОДАТОК Б

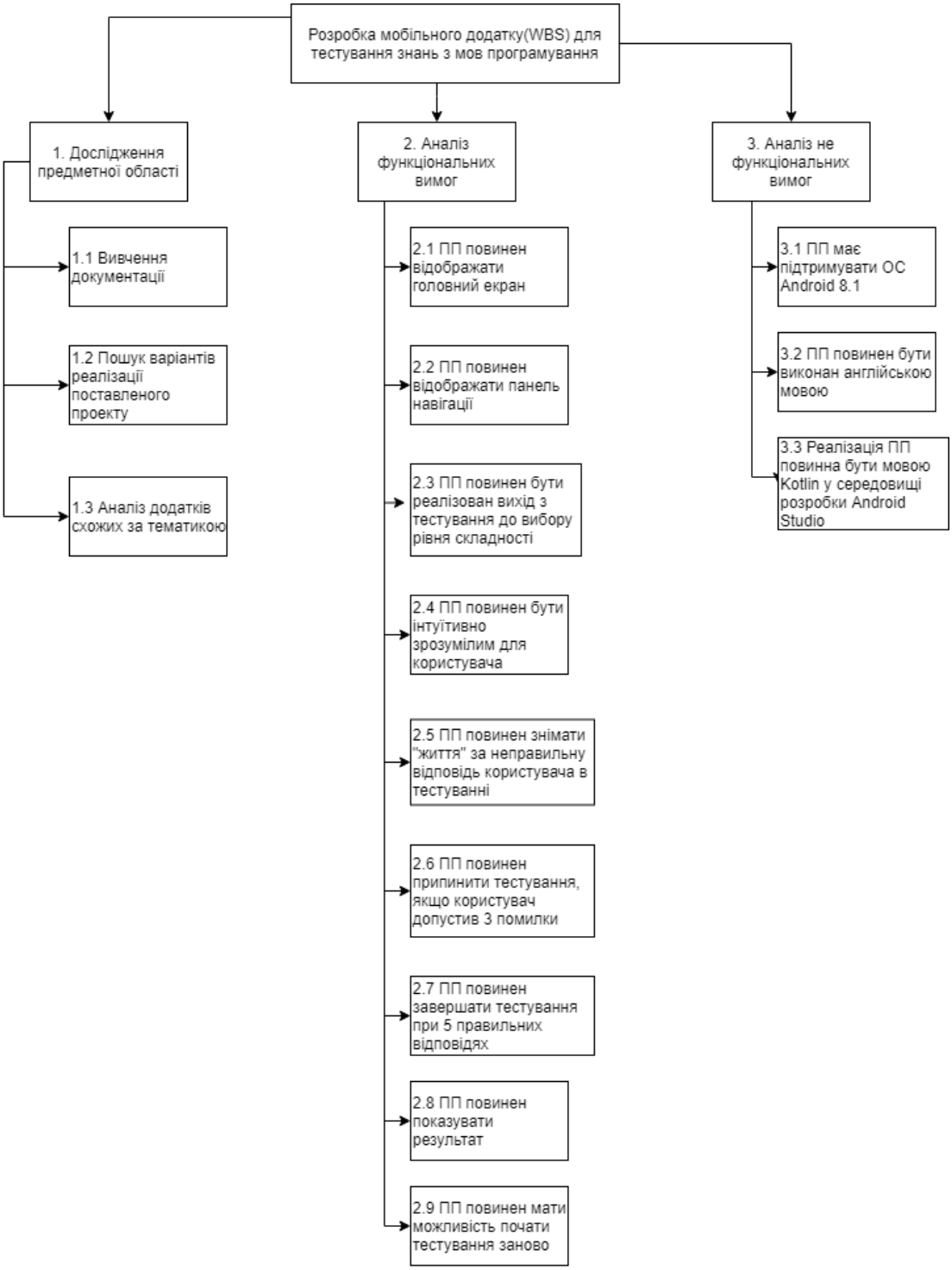
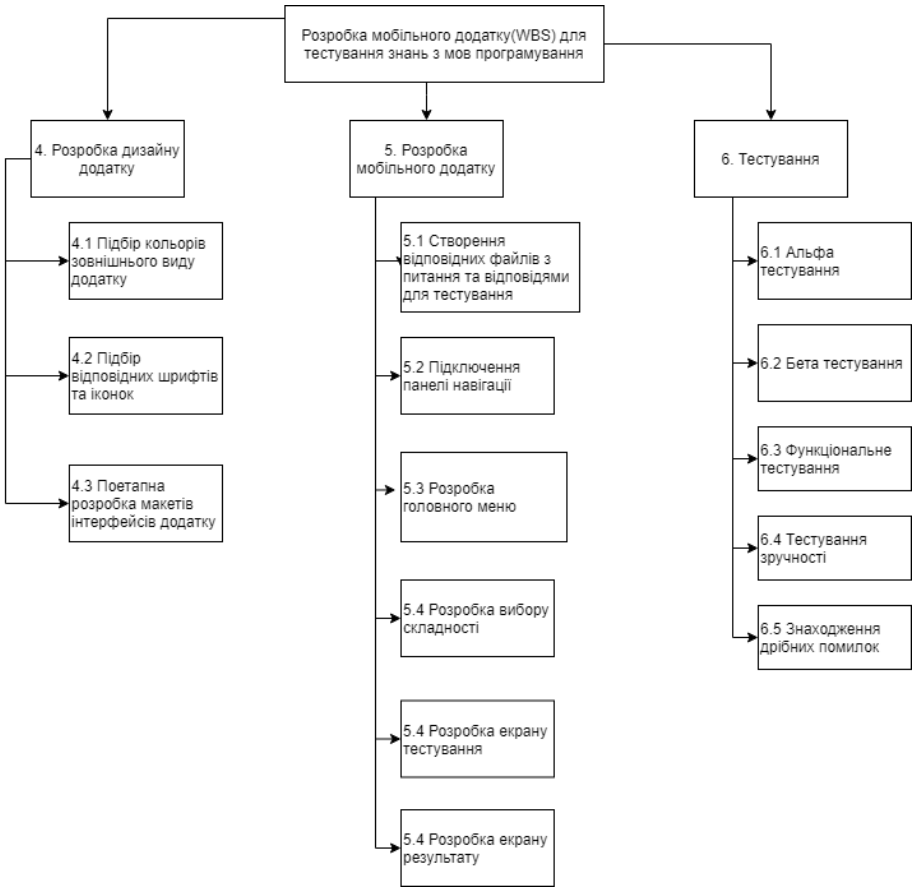


Схема Б.1 – Планування змісту структури мобільного додатку(WBS) длятестування знань з мов програмування



Продовження схеми Б.1 - Планування змісту структури мобільного додатку для тестування знань з мов програмування (WBS)

Продовження додатку Б

1. Вивчення документації ТЗ	+
1.1 Пошук варіантів реалізації поставленого проекту	+
1.2 Вивчення документації ТЗ	+
1.3 Ознайомлення з контентом, який буде надано користувачу з додатку	+
2. Аналіз функціональних вимог	+
2.1 Користувачу потрібно зайти в свій профіль або зареєструватися	+
2.2 При відновленні профілю Користувачупотрібно вказати у екрані відновлення паролю свою пошту, яка вже буда зареєстрована в системі	+
2.3 Мобільний додаток повинен на запиткористувача відправити на пошту інструкцію відновлення паролю	+
2.4 Користувач потрібно перейти запосиланням та вказати новий пароль для свого профілю щоб відновити доступ до додатку	+
2.5 ПП повинен відображати головний екран після проходження аутентифікації	+
2.6 ПП повинен відображати панель навігації	+
2.7 У ПП повинен бути передбачена вихід з тестування	+
2.8 Користувацький інтерфейс ПП повиненбути інтуїтивно зрозумілим для користувача	+

Таблиця Б.2 – Матриця відповідальності

Продовження таблиці Б.2 – Матриця відповідальності

2.10 У ПП повинно знімати “ життя ” якщо користувач не правильно відповідає на питання тесту:	+
2.11 У ПП повинно припиняти тестування якщо користувач відповів не правильно 3 рази	+
2.12ПП повинний показувати результат проходження тесту	+
3. Аналіз не функціональних вимог	+
3.1 ПП маж підтримувати ОС Android 8.1	+
3.2 ПП повинен бути виконаний українською мовою	+
3.3 Реалізація ПП відбувається Java у середовищі розробки Android Studio	+
4. Розробка концепції дизайну	+
4.1 Підбір кольорів зовнішнього виду додатку	+
4.2 Розробка макетів інтерфейсів додатку	+
5. Розробка мобільного додатку	+
5.1 Створення відповідних файлів з питання та відповідями для тестування	+
5.2 Розробка системи аутентифікації	+
5.3 Верстка головних екранів	+
5.5 Підключення панелі навігації та вибір відповідного тесту	+
5.6. Розробка системи тестування	+
5.7. Розробка фінального результату тестування	+
5.8. Встановлення взаємодії всього функціоналу додатка	+

6. Тестування	+
6. 1 α-тестування	+
6.2β-тестування	+
6.3Тестування зручності	+
6.4функціональне тестування	+
6.5Знаходження дрібних помилок	+
6.6знаходження дрібних помилок	+
6.7Виправлення помилок	+

Продовження таблиці Б.2 – Матриця відповідальності

Таблиця Б.3 – Ймовірність втрат

Ризик	Ймовірність виникнення	Величина втрат
Не ефективне використання робочим часом	2	3
Втрачання мотивації команди	2	4
Непорозуміння з замовником	3	4
Низька продуктивність роботи	3	3
Недоречне використання ресурсів	3	3
Не доцільно складене ТЗ	2	4
Неефективний розподіл робіт	3	4
Недостатній професіоналізм	3	4
Внесення змін у ТЗ	3	4
Виникнення надзвичайний ситуацій	2	4

ПРОДОВЖЕННЯ ДОДАТКУ Б

Таблиця Б.4 – Класифікація за ступенем впливу та за рівнем ризику

Ризик	Ступінь впливу	Рівень ризику
Не ефективне використання робочим часом	13	виправданні ризики
Втрачання мотивації команди	12	виправданні ризики
Непорозуміння з замовником	18	виправданні ризики
Низька продуктивність роботи	15	виправданні ризики
Недоречне використання ресурсів	12	виправданні ризики

Таблиця Б.4 – Класифікація за ступенем впливу та за рівнем ризику

Продовження додатку Б

Продовження таблиці Б.4 – Класифікація за ступенем впливу та за рівнем ризику

Не доцільно складене ТЗ	15	недопустимі ризики
Неефективний розподіл робіт	12	недопустимі ризики
Недостатній професіоналізм	17	недопустимі ризики
Внесення змін у ТЗ	20	виправданні ризики

ДОДАТОК В

MainActivity.kt

```
class MainActivity : AppCompatActivity() {



    /**
     * Method that overrides default version of onKeyDown method.
     * When back button is pressed, the application returns to the first screen.
     * @param keyCode Int
     * @param event KeyEvent
     * @return Boolean
     */
     override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
        if(keyCode == KeyEvent.KEYCODE_BACK) {
            val intent = Intent(this, MainActivity::class.java)
            finish()
            startActivity(intent)
        }
        return true
    }
    /**
     * Private method called with start of MainActivity class.
     * Calls super method onCreate, sets content view and calls method responsible for
     * configuring buttons.
     * @param savedInstanceState Parameter of Bundle? type, needed to call super method.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        set_buttons()
    }

    /**
     * Method that configures all buttons and onClickListener for first 2 views.
     */
    private fun set_buttons()
    {
        btn_start_game.setOnClickListener{
            setContentView(R.layout.wybor_kategorii)
            val button_hard = btn_hard
            button_hard.setOnClickListener {
```

```
                val kat = "hard"
                val intent = Intent(this,GameActivity::class.java)
                intent.putExtra("kat", kat)
                startActivity(intent)
            }
            val button_middle = btn_middle
            button_middle.setOnClickListener {
                val kat = "middle"
                val intent = Intent(this,GameActivity::class.java)
                intent.putExtra("kat", kat)
                startActivity(intent)
            }
            val button_easy = btn_easy
            button_easy.setOnClickListener {
                val kat = "easy"
                val intent = Intent(this,GameActivity::class.java)
                intent.putExtra("kat", kat)
                startActivity(intent)

            }

        }

    }

}
```

Words.kt

```
class Words {
    /**
     * Private method used to read all words from a file into an array.
     * @param context Context type parameter, needed to use AssetManager.
     * @param fileName String with name of file to open
     * @return Returns array of words from file.
     */
     fun readFile(context: Context, fileName: String): Array<String> {
        val am: AssetManager = context.assets
        val array = Array<String>(5){"it =$it"}
        val `is`: InputStream = am.open(fileName)
        val buffreader = BufferedReader(`is`.reader())
        val content = StringBuilder()
```

```
        var i=0
        try {
            var line = buffreader.readLine()
            while (line != null && i<5) {
                array.set(i, line)
                content.append(line + "\n")
                line = buffreader.readLine()
                i++
            }
        } finally {
            buffreader.close()
        }
       return array
    }

    /**
     * Method used to get random word.
     * @param context Context type parameter. .
     * @param fileName String with name of file to open
     * @return Returns random word from specified file.
     */
    fun get_rand_word( context: Context, fileName: String): String
    {
        val array = readFile(context,fileName)
        val rand = Random()
        val x: Int = rand.nextInt(5)
        return array[x]
    }

    /**
     * Method used to get random word other than that in paramter, without reading file
again.
     * @param good_word String with a word assumed to be good. Needed to randomly get only
wrong
     * images.
     * @param array Array of String used to get names of pictures.
     * @return Returns random word.
     */
    fun get_rand_words(good_word: String, array: Array<String>): String
    {
        val rand = Random()
```

```
        var x: Int = rand.nextInt(5)
        while(true)
            if(array[x].equals(good_word)) {
                x = rand.nextInt(5)
            }else {
                break
            }
        return array[x]
    }

}
```

TheEndActivity.kt

```
class TheEndActivity: AppCompatActivity()  {
        override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
        if(keyCode == KeyEvent.KEYCODE_BACK) {
            val intent = Intent(this, MainActivity::class.java)
            finish()
            startActivity(intent)
        }
        return true
    }
        override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.the_end)
        val Nol = intent.getIntExtra("amount_of_lives", -1)
        val Ga = intent.getIntExtra("goodans", -1)
        val Wa = intent.getIntExtra("wrongans", -1)

        if (Nol == 0)
            gratulacje.text = resources.getText(R.string.bedzie_lepiej)
        else
            gratulacje.text = resources.getText(R.string.gratulacje)
        numberoflives.text = Nol.toString()
        goodanserws.text = Ga.toString()
        wronganserws.text = Wa.toString()
        game_again.setOnClickListener {
            startActivity(Intent(this@TheEndActivity, MainActivity::class.java))  }
        }
}
```

GameActivity.kt

```
class GameActivity : AppCompatActivity() {

    private var imageResource: Int = 0
    private var ximageResource: Int = 0
    private var yimageResource: Int = 0
    private var zimageResource: Int = 0
    private var amount_of_lives_counter = 3
    private val pics = arrayOfNulls<ImageView>(4)


    /**
     * Method that overrides default version of onKeyDown method.
     * When back button is pressed, the application returns to the first screen.
     * @param keyCode Int
     * @param event KeyEvent
     * @return Boolean
     */
    override fun onKeyDown(keyCode: Int, event: KeyEvent?): Boolean {
        if(keyCode == KeyEvent.KEYCODE_BACK) {
            val intent = Intent(this, MainActivity::class.java)
            finish()
            startActivity(intent)
        }
        return true
    }
    /**
     * Private method used to configure home button.
     */
    private fun setHomeButton()
    {
        val button_pow = btn_powrot
        button_pow.setOnClickListener {
            val intent = Intent(this,MainActivity::class.java)
            startActivity(intent)

        }
    }
    /**
     * Method called with start of GameActivity class.
     * Calls super method onCreate and controls the main part of the game.
     * @param savedInstanceState Parameter of Bundle? type, needed to call super method.
     */
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_game)
        setHomeButton()
        val good_answers_counter = 0
        val wrong_answers_counter = 0
        amount_of_lives.text = amount_of_lives_counter.toString()
        good_answers.text = good_answers_counter.toString()
        wrong_answers.text = wrong_answers_counter.toString()
        val kat: String?= intent.getStringExtra("kat")
        losuj(kat, good_answers_counter, wrong_answers_counter)
    }

    /**
     * Private Method that sets imageViews with images described by parameters wrong_images
and good_word.
```

```
     * @param wrong_images Array of Strings with randomly chosen words for wrong images.
     * @param good_word String with randomly chosen word that is correct.
     */
    private fun setImages(wrong_images: Array<String>, good_word: String): Int
    {
        val wrong_image1 = wrong_images[0]
        val wrong_image2 = wrong_images[1]
        val wrong_image3 = wrong_images[2]
        imageResource = resources.getIdentifier("@drawable/$good_word", null,
this.packageName)
        ximageResource = resources.getIdentifier("@drawable/$wrong_image1", null,
this.packageName)
        yimageResource = resources.getIdentifier("@drawable/$wrong_image2", null,
this.packageName)
        zimageResource = resources.getIdentifier("@drawable/$wrong_image3", null,
this.packageName)
        val resources: IntArray = intArrayOf(imageResource, ximageResource, yimageResource,
zimageResource)
        pics[0] = imageView1
        pics[1] = imageView2
        pics[2] = imageView3
        pics[3] = imageView4

        val used_pics = arrayOfNulls<Int>(4)
        val used_resources = arrayOfNulls<Int>(4)
        var pics_values : Int
        var goodImage = 0
        var resource_values : Int
        var i = 0

        while(i<4)
        {
            pics_values = Random.nextInt(0, 100)
            resource_values = Random.nextInt(0, 100)
            if ((pics_values % 4) in used_pics)
            {
                continue
            } else if ((resource_values % 4) in used_resources)
            {
               continue
            } else
            {
                used_pics[i] = pics_values % 4
                used_resources[i] = resource_values % 4
                pics[pics_values%4]?.setImageResource(resources[resource_values%4])
                i++
                if ((resource_values % 4) == 0)
                    {
                        goodImage = pics_values % 4
                    }
            }
        }

        return goodImage
    }

    /**
     * Private method used to configure what will happen when right or wrong image is
clicked.
     * @param number_good_ans Integer that keeps number of good answers given by user.
     * @param number_wrong_ans Integer that keeps number of wrong answers given by user.
     * @param kat String that keeps name of category chosen by user in previous activity.
```

```
     */
    private fun setOnClicks(number_good_ans: Int, number_wrong_ans: Int, kat: String?,
goodImage: Int)
    {
        var good_answers_counter = number_good_ans
        var wrong_answers_counter = number_wrong_ans
        var imagebad1 = 0; var imagebad2 = 0; var imagebad3=0
        if (goodImage == 0)
        {
            imagebad1 = 1
            imagebad2 = 2
            imagebad3 = 3
        } else if (goodImage == 1)
        {
            imagebad1 = 0
            imagebad2 = 2
            imagebad3 = 3
        }
        else if (goodImage == 2)
        {
            imagebad1 = 0
            imagebad2 = 1
            imagebad3 = 3
        }
        else if (goodImage == 3)
        {
            imagebad1 = 0
            imagebad2 = 1
            imagebad3 = 2
        }
        pics[goodImage]?.setOnClickListener {
            good_answers_counter += 1
            good_answers.text = good_answers_counter.toString()
            losuj(kat, good_answers_counter, wrong_answers_counter)
        }
        pics[imagebad1]?.setOnClickListener {
            wrong_answers_counter += 1
            wrong_answers.text = wrong_answers_counter.toString()
            amount_of_lives_counter -= 1
            if (amount_of_lives_counter == -1)
            {
                amount_of_lives.text = "0"
                amount_of_lives_counter = 0
                endGame(good_answers_counter, wrong_answers_counter)
            }
            else
            {
                amount_of_lives.text = amount_of_lives_counter.toString()
                losuj(kat, good_answers_counter, wrong_answers_counter)
            }

        }
        pics[imagebad2]?.setOnClickListener {
            wrong_answers_counter += 1
            wrong_answers.text = wrong_answers_counter.toString()
            amount_of_lives_counter -= 1
            if (amount_of_lives_counter == -1)
            {
                amount_of_lives.text = "0"
                amount_of_lives_counter = 0
                endGame(good_answers_counter, wrong_answers_counter)
            }
```

```
            else
            {
                amount_of_lives.text = amount_of_lives_counter.toString()
                losuj(kat, good_answers_counter, wrong_answers_counter)
            }
        }
        pics[imagebad3]?.setOnClickListener {
            wrong_answers_counter += 1
            wrong_answers.text = wrong_answers_counter.toString()
            amount_of_lives_counter -= 1
            if (amount_of_lives_counter == -1)
            {
                amount_of_lives.text = "0"
                amount_of_lives_counter = 0
                endGame(good_answers_counter, wrong_answers_counter)
            }
            else
            {
                amount_of_lives.text = amount_of_lives_counter.toString()
                losuj(kat, good_answers_counter, wrong_answers_counter)
            }
        }
    }

    /**
     * Private method used to set variables for the game. Checks if it should end the game
based
     * on amount of lives and good answers given
     * @param kat String that keeps name of category chosen by user in previous Activity.
     * @param goodanswers Integer that keeps number of good answers given by user.
     * @param wronganswers Integer that keeps number of wrong answers given by user.
     */
    private fun losuj(kat: String?, goodanswers: Int, wronganswers: Int) {
        if (isEndOfGame(goodanswers))
        {
            endGame(goodanswers, wronganswers)
        } else {
            val word = Words()
            val good_word = onClickTemp(word, kat)
            val wrong_images = Array<String>(3) { "it =$it" }
            var str : String
            var i = 0
            while(true)
            {
                str = word.get_rand_words(good_word, word.readFile(this,"slowka/" + kat +
".txt" ))
                if (str != wrong_images[0] && str != wrong_images[1] && str !=
wrong_images[2]   )
                {
                    wrong_images[i] = str
                    i++
                }
                if (i>2)
                    break
            }
            val goodImage = setImages(wrong_images, good_word)
            if(good_word.contains('_'))
            {
                slowko.text = good_word.replace('_', ' ')
            }
            else
            {
```

```
                slowko.text = good_word
            }
            setOnClicks(goodanswers, wronganswers, kat, goodImage)
        }

    }

    /**
     * Private method called when game is supposed to end. Starts TheEndActivity.
     * @param goodanswers Integer that keeps number of good answers given by user.
     * @param wronganserws Integer that keeps number of wrong answers given by user.
     */
    private fun endGame(goodanswers: Int, wronganserws: Int)
    {
        intent = Intent(this, TheEndActivity::class.java)
        intent.putExtra("amount_of_lives",amount_of_lives_counter)
        intent.putExtra("goodans",goodanswers)
        intent.putExtra("wrongans",wronganserws)
        startActivity(intent)
    }

    /**
     * Private method that checks if game should end.
     * @return Returns True if user has no lives left or has given 10 good answers,
     * and False otherwise.
     */
    private fun isEndOfGame(goodanswers: Int):Boolean
    {
        if (amount_of_lives_counter < 0 || goodanswers == 3)
            return true

        return false
    }

    /**
     * Private method that gets random word from category chosen by user.
     * @param word Object of Words class, used to find random word.
     * @param kat String that keeps name of category chosen by user
     * @return Returns word from chosen category.
     */
    private fun onClickTemp(word: Words, kat: String? ): String
    {
        return word.get_rand_word(this, "slowka/"+kat+".txt")
    }
}
```

Додаток Ж

Taras Shevchenko National University of Kyiv

Development of an adaptive distance learning system in the C # programming language

Software Architecture Document (SAD)

Content Owner: Dmytro Kupin

DOCUMENT NUMBER: 1.0

RELEASE: 1.0

RELEASE DATE: 30.05.2021

TABLE OF CONTENTS

1. Introduction

1.1. Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system.

1.2. Scope

This Software Architecture Document provides an architectural overview of the Software for learning programming language C#. The software provides user with an easy to use learning application in test format.

1.3. Definitions

Programming learning app – is an application for mobile devices used to improve user`s programming language knowledge.

1.4. References

Applicable references are:

- Programming learning app
- Programming test
- Kotlin
- Android
- Android Studio

2. Architecture Representation

This document presents the architecture as a series of views; use case view, sequence view, component view. Software system contains an option to choose between different levels of difficulty to make it more flexibile to use and to be adaptive for any user.

3. Architectural Goals and Constraints

There are some key requirements and system constraints.

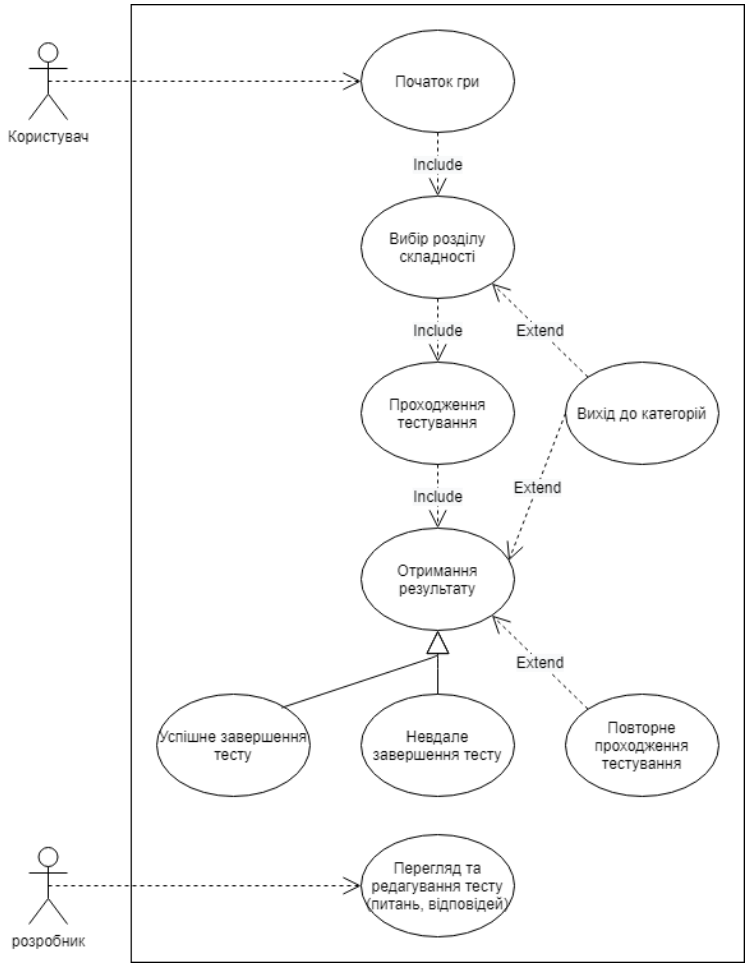
- Test answer input must be in real-time.

- Test must contain 3 levels of difficulty.
- Test result window must be configured in real-time.
- Applitcation must be easy to understand in terms of navigation.

4. Use-case View

A usage diagram is a list of actions, a scenario in which a user interacts with an application or program to perform an action to achieve a specific goal.

The use cases were indicated in the text and are represented by a diagram of use cases. The essence component of the interaction diagram modeling helps to develop the system from the point of view of the future user. Using precedent, you can describe the requirements of the user, the requirements for interaction with the system.



Programming system has types of use-cases:

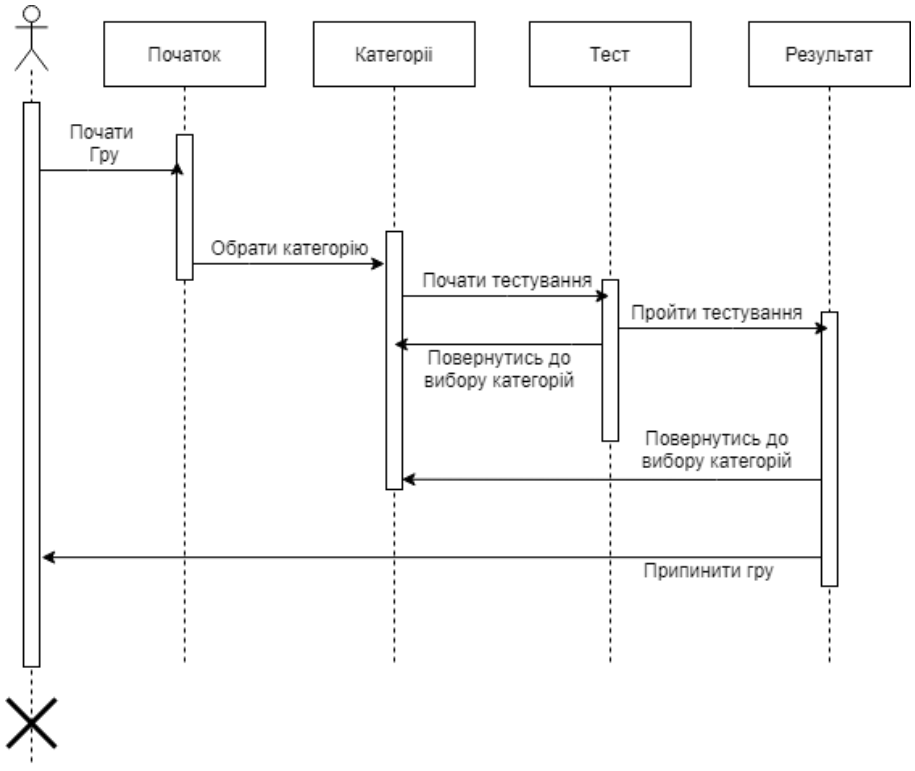
- Categories – user choose the difficulty level of the test.

- Testing – starting test, answering the questions and receiving result after completing.
- Return to categories – if user needs to start test with another difficulty level.

5. Sequence View

Sequence diagram is a type of diagram in UML. The sequence diagram shows the interactions of objects ordered in time. In particular, such diagrams show the objects involved and the sequence of messages sent.

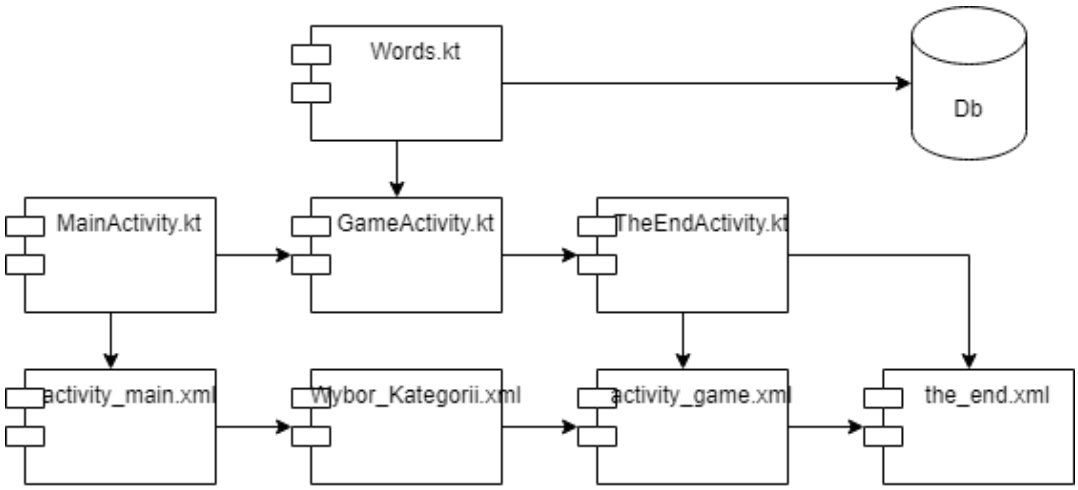
The sequence view of the program is comprised of the 4 main parts: beginning, categories, test, result.



6. Component View

The component diagram describes the physical representation of the system and provides the transition from logical representation to project implementation in the form

of program code. A component is part of the physical implementation of a system (such as a software module or user interface) that encapsulates a set of functionalities.



7. Size and Performance

The chosen software architecture supports the key sizing and timing requirements:

- The system shall support only 1 user per one test-session.
- The test answer input shall be displayed in real-time with no more than a 1 second latency.
- The test result shall be displayed in real-time with no more than a 1 second latency after completing.